УДК 004

РЕАЛИЗАЦИЯ КОНТРАКТНОГО ТЕСТИРОВАНИЯ МИКРОСЕРВИСОВ

Ореховский А. Университет ИТМО
Научный руководитель – кандидат технических наук, доцент Маркина Т. А.
Университет ИТМО

Введение. В последнее время в сфере разработки веб приложений стал популярен микро-сервисный подход. При тестировании продуктов с таким подходом часто используют сценарные тесты [1], в которых проверяется интеграция различных компонент системы на специально выделенном окружении. Такие тесты на живых сервисах дают самый достоверный результат, но в то же время они — одни из самых дорогих. При этом, данные тесты нестабильны и занимают больший объем времени в сравнении с другими видами тестирования. Уменьшить издержки тестирования может помочь контрактное тестирование — подход, при котором сервисы, обменивающиеся информацией, закрепляют список используемых данных и методы их получения в так называемом контракте. Это помогает сервисам развиваться, зная, что задокументированные данные не изменят свой тип или структуру [2].

Основная часть. Для реализации контрактного тестирования необходимы следующие элементы:

- язык описания контрактов, который позволяет сервисам-потребителям описывать данные и способы их получения на основе спецификаций сервисов-поставщиков;
- хранилище контрактов и спецификаций;
- алгоритмы сопоставления контрактов спецификациям для валидации того, что контракт основан на реальном поведении сервисов-поставщиков;
- алгоритмы версионирования спецификаций для установления того, что новые версии спецификаций не конфликтуют с предыдущими версиями.

Данный подход не является новым, и уже существует немалое количество библиотек и фреймворков, самым популярным из которых является Pact [3], реализующих механизмы контрактного тестирования, хотя не все из них достаточно гибкие, чтобы удовлетворить условия стимуляции обновления потребителей версий используемых программных интерфейсов и написания контрактов, ориентированных на поставщика. Последний пункт подразумевает написание контрактов основываясь на спецификациях существующих сервисов. Учитывая, что аналоги не удовлетворяют данным условиям, необходимость создания новой библиотеки контрактного тестирования считается обоснованной.

Проблемой так же является то, что, однажды использовав функционал поставщика, код сервисов-потребителей не меняется, из-за чего могут возникнуть уязвимости в системе без должного коммуницированния команд разработки. Форсировать переход кодовой базы потребителей на новые версии сервисов-поставщиков может помочь интеграция конвейера постоянного обновления и поставки кода на стороне сервисов-потребителей в сервис, хранящий спецификации и контракты, например при помощи триггеров [4].

Выводы. Учитывая вышеперечисленные условия и основываясь на анализе аналогов контрактного тестирования, был разработан прототип, позволяющий сравнивать схемы поставщиков и контракты, составленные потребителями. Дальнейшая разработка ориентирована на создание сервиса, использующего данный прототип для хранения и сопоставления контрактов и схем, который позволит провоцировать потребителей на переход на более новые версии программных интерфейсов за счет интеграции с элементами непрерывной интеграции и доставки, при этом контракты будут ориентированы на потребителя.

Список использованных источников:

1. Паттерн: Контрактные Тесты Сервиса Интеграций - URL: https://microservices.io/patterns/testing/service-integration-contract-test.html
2. Л. Криспин, Д. Грегори Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд – 2016 г.
3. Фреймворк контрактного тестирования Pact – URL: https://docs.pact.io/.
4. Триггеры в системе постоянной интеграции и доставки GitLab – URL: https://docs.gitlab.com/ee/ci/triggers/